Optimistic planning for question selection

Anonymous Author(s)
Affiliation
Address
email

Abstract

In many online educational environments, a common problem is how to select the best questions to give to a user in order to maximize their learning over the limited period of time they interact with the platform. We consider the problem of determining the best policy, or sequence of questions, to give to a user in a fixed period where both the amount of time they spend on each question and the benefit they gain from it are stochastic (but can be simulated). This problem is, in essence, the stochastic knapsack problem, a PSPACE-hard problem that has been studied extensively in operations research. It is desirable that any solution algorithm produces sequences of question that depend on the amount of time that the user spent on the preceding questions. We propose a new adaptive algorithm for the stochastic knapsack problem that uses techniques from multi-armed bandits to explore only potentially good policies. Our algorithm is adaptive to the amount of remaining time and can be shown to obtain a solution within ϵ of the optimal. We then discuss the use of this algorithm in an online education environment.

1 Introduction

In online education, one key challenge is to provide students with personalized sequences of questions that make the best use of the users time to maximize learning. Imagine an app that a user interacts with for 15 minutes a day in order to learn a new language or skill. The app must decide which questions it can give the student in this time period in order to maximize their learning. However, the amount of time the student will take to complete each question and the benefit they will gain from doing so are both stochastic. This stochasticity makes the problem considerable more difficult. In order to deal with it, we assume we are able to simulate accurately user time and reward from completing each question. Much work has focused on obtaining good predictive models of student performance and question time (see for example, [7, 11, 17]), and we assume we have access to such models at an individual student level (since this leads to better decision making [13]).

The stochastic knapsack problem [8], is a classic resource allocation problem that consists of selecting a subset of items to place into a knapsack of a given capacity. Placing each item in the knapsack consumes a random amount of the capacity and provides a stochastic reward. Our problem of selecting which questions to give to a user in a 15 minute exercise can be thought of as the stochastic knapsack problem. Each question (item) will take a random amount of time (size) and improve the student's knowledge in a stochastic manner (reward). To make optimal use of the available time the app needs to track the progress of the user and adjust accordingly. Once an item is placed in the knapsack, we assume we observe its realized size and can use this to make future decisions. This enables us to consider adaptive or closed loop strategies, which will generally perform better [9] than open loop strategies in which the schedule is invariant of the remaining time.

For our purposes, it is desirable to have methods for the stochastic knapsack problem that can make use of all available resources and adapt with the remaining capacity. One manner of obtaining such adaptive solutions is to model the problem as a decision tree as discussed in [9]. However, in all

Submitted to 29th Conference on Neural Information Processing Systems (NIPS 2016). Do not distribute.

but the simplest cases, this decision tree will be too large to search exhaustively. We propose using ideas from optimistic planning [4, 16] to significantly accelerate the tree search by only considering policies with high upper confidence bounds. Most optimistic planning algorithms were developed for discounted mdps and as such rely on discount factors to limit future reward which are not present in the stochastic knapsack problem. Furthermore, in our problem, the random variable representing state transitions also provides us with information on the future rewards. To avoid discount factors and to use the transition information, we work with confidence bounds that incorporate estimates of the remaining capacity and use these estimates to determine how many samples we need. In order to do this, we need techniques that can deal with weak dependencies and that give us confidence regions that hold simultaneously for multiple sample sizes. We therefore combine Doob's martingale inequality with Azuma-Hoeffding bounds to create high probability bounds. Following the optimistic planning approach, we use these bounds to develop an algorithm that adapts to the complexity of the problem instance: it is guaranteed to find an ϵ-good approximation independent of how difficult the problem is and, if the problem instance is easy to solve, it expands only a moderate sized tree.

1.1 Related work

Finding exact solutions to the simpler deterministic knapsack problem, in which item weights and rewards are deterministic, is known to be NP-hard and it has been stated that the stochastic knapsack problem is PSPACE-hard [9]. Therefore, most work on the stochastic variant of the problem has focused on approximations. The state-of-the-art approaches to the stochastic knapsack problem where the reward and resource consumption distributions are known, were introduced in [9] where the authors introduced a heuristic that is adaptive and comes within a factor of a $1/3$rd of the best total reward. The heuristic groups the available items into small and large items and fills the knapsack exclusively with items of one of the two groups. The strategy for small items is non-adaptive and orders these items according to their reward - consumption ratio, placing items into the knapsack according to this ordering. For the large items, a decision tree is built to some predefined depth d and an exhaustive search for the best policy in that decision tree is performed.

Optimistic planning was developed for tree search in large deterministic [12] and stochastic (both open [3] and closed [4] loop) systems. The general idea is to use the upper confidence principle of the UCB algorithm for multi-armed bandits [1] to expand a tree. This is achieved by expanding nodes that have the potential to lead to good solutions through using bounds that take into account both the reward received in getting to a node and the reward that could be obtained after moving on from that node. [16] use optimistic planning in discounted MDPs, requiring only a generative model of the rewards and transitions. Instead of the UCB algorithm, their work relies on the best arm identification algorithm of [10]. Optimistic planning algorithms are used to return a near optimal first action and are then rerun to select the next action. In our case, the decision tree is a good approximation to the entire problem so we can output a near-optimal policy. In our problem, the state transitions (size of item/time taken to answer question) provide information about future rewards and so should be considered when defining the high confidence bounds. Furthermore, our algorithm iteratively builds confidence bounds which are used to determine whether it is necessary to sample more thus making more effective use of resources. One would imagine that the `StOP` algorithm from [16] could be easily adapted to the stochastic knapsack problem. However, the assumptions required for this algorithm to terminate are too strong for it to be considered a feasible algorithm for our problem.

In the education literature, there has been some work done on the question selection problem. [6] propose to use a non-stationary multi-armed bandit algorithm to select questions for students, without the time constraints that are present in our problem. [15] consider the use of POMDPs to determine which type of action to take next based on previous success but focus on the choice between types of actions (such as videos, quizes or questions with feedback) rather than the specific question itself. [5] also use reinforcement learning algorithms to work towards a similar aim.

1.2 Our contribution

Our main contributions are the confidence bounds in Lemma 1 and Proposition 2 that allow us to simultaneously estimate remaining capacity and reward, with guarantees that hold uniformly over multiple sample sizes; Proposition 3, which shows that we can avoid discount based arguments and still return an adaptive policy with value within ϵ of the optimal policy with high probability while using varying capacity estimates; and, primarily, our algorithm `OpStoK` whose use to provide instructional policies for education software will be discussed in Section 5.

2 Problem formulation

In this section, we formally define our problem. Note that since our problem is essentially the stochastic knapsack problem, we define it in terms of the knapsack definitions and describe it using items, sizes and budgets, rather than questions, time taken and time limits.

We consider the problem of selecting a subset of items from a set of K items, I, to place into a knapsack of capacity B where each item can be played at most once. For each item $i \in I$, let C_i and R_i be bounded random variables defined on a joint probability space $(\Omega, \mathcal{A}, P)$ which represent the size and reward of item i. It is assumed that we can simulate from the generative model of (R_i, C_i) for all $i \in I$ and we will use lower case c_i and r_i, to denote realizations of the random variables. We assume that the random variables (R_i, C_i) are independent of (R_j, C_j) for all $i, j \in I$, $i \neq j$. Further, it is believed that item sizes and rewards do not change dependent on the other items in the knapsack. We assume the problem is non-trivial, in the sense that it is not possible to fit all items in the knapsack at once. If we place an item i in the knapsack and the consumption C_i is strictly greater than the remaining capacity then we gain no reward for this item. Our final important assumption is that there exists some non-decreasing function $\Psi(\cdot)$, satisfying $\lim_{b \to 0} \Psi(b) = 0$ and $\Psi(B) < \infty$, such that the reward that can be achieved with budget b is upper bounded by $\Psi(b)$.

Representing the stochastic knapsack problem as a tree requires that all item sizes take discrete values. While in this work, it will generally be assumed that this is the case, in some problem instances, continuous item sizes need to be discretized. In this case, let v^* be the optimal value of the best policy and let ξ^* be the corresponding discretization error. Then $\Psi(\xi^*)$ is an upper bound on the extra reward that could be gained from the space lost due to discretization. For discrete sizes, we assume there are s possible values the random variable can take and that there exists a value $\theta > 0$ such that $C_i \geq \theta$ for all $i \in I$.

2.1 Planning trees and policies

The stochastic knapsack problem can be thought of as a planning tree with the initial empty state as the root at level 0. Each node on an even level is an *action* node and its children represent placing an item in the knapsack. The nodes on odd levels are *transition* nodes with children representing item sizes. We define a *policy* Π as a finite subtree where each action node has at most one child and each transition node has s children. The *depth* of a policy Π, $d(\Pi)$, is defined as the number of transition nodes in any realization of the policy (where each transition node has one child), or equivalently, the number of items. Let $d^* = \lfloor B/\theta \rfloor$ be the maximal depth of any policy. For any $1 \leq d \leq d^*$, the number of policies of depth d is

$$N_d = \prod_{i=0}^{d-1} (K - i)^{s^i} \tag{1}$$

where $K = |I|$ is the number of items, and s the number of discrete sizes.

We define a *child* policy, Π', of a policy Π as a policy that follows Π up to depth $d(\Pi)$ then plays additional items and has depth $d(\Pi') = d(\Pi) + 1$. In this setting, Π is the *parent* policy of Π'. A policy is said to be *incomplete* if the remaining budget allows for another item to be inserted into the knapsack (see Section 4 for a formal definition).

The *value* of a policy Π can be defined as the cumulative expected reward obtained by playing items according to Π, $V_\Pi = \sum_{t=1}^{T} E[R_{i_t}]$ where i_t is the t-th item chosen by Π. Let $\mathcal{P}$ be the set of all policies, then define the *optimal policy* as $\Pi^* = \arg\max_{\Pi \in \mathcal{P}} V_\Pi$, and corresponding *optimal value* as $v^* = \max_{\Pi \in \mathcal{P}} V_\Pi$. Our algorithm returns an ϵ-optimal policy with value $v^* - \epsilon$. For any policy Π, we define a *sample* of Π as follows. The first item of any policy is fixed so we take a sample of the reward and size from the generative model of that item. We then use Π to tell us which item to sample next (based on the size of the previous item) and sample the reward and size of that item. This continues until the policy finishes or the cumulative size of the selected items exceeds B.

3 High confidence bounds

In this section, we develop confidence bounds for the value of a policy. Observe that a policy Π need not consume all available budget, in fact our algorithm will construct iteratively longer policies starting from the shortest policies of playing a single item. Consequently, we are also interested in R^+_Π, the expected maximal reward that can be obtained after playing according to policy Π until all the budget is consumed. Let B_Π be a random variable representing the remaining budget after playing according to a policy Π. Our assumptions guarantee that there exists a function Ψ such that $R^+_\Pi \leq E\Psi(B_\Pi)$. We define V^+_Π to be the maximal expected value of any continuation of policy Π so $V^+_\Pi = V_\Pi + R^+_\Pi \leq V_\Pi + E\Psi(B_\Pi)$.

From m samples of the reward of policy Π, we estimate the value of Π as $\overline{V_\Pi}_m = \frac{1}{m}\sum_{j=1}^{m}\sum_{d=1}^{d(\Pi)} r^{(j)}_{i(d)}$, where $r^{(j)}_{i(d)}$ is the reward of item $i(d)$ chosen at depth d of sample j. However, our real interest is in the value of V^+_Π since we wish to identify the policy with greatest reward when continued until the budget is exhausted. From Hoeffding's inequality, $P\left(|\overline{V_\Pi}_{m_1} - V^+_\Pi| > E\Psi(B_\Pi) + \sqrt{\frac{\Psi(B)^2\log(2/\delta)}{2m}}\right) \leq \delta$. This bound depends on the quantity $E\Psi(B_\Pi)$ which is typically not known. Furthermore, our algorithm will work by sampling $\Psi(B_\Pi)$ until we are confident enough that it is small or large. As such, it introduces weak dependencies into the sampling process and we need confidence bounds that will hold simultaneously for multiple numbers of samples of the remaining capacity, m_2. Hence, we work with martingale techniques and use Azuma-Hoeffding like bounds [2], similar to the technique used in [14]. Specifically, in Lemma 3 (supplementary material), we use Doob's maximal inequality and a peeling argument to get Azuma like bounds for the maximal deviation of the sample mean from the expected value under boundedness assumptions. Assuming we sample the reward m_1 times and remaining capacity of a policy $m_2 \leq n$ times, the following key result holds.

Proposition 1 *The Algorithm `BoundValueShare` (Algorithm 2) returns confidence bounds,*

$$L(V^+_\Pi) = \overline{V_\Pi}_{m_1} - c_1$$
$$U(V^+_\Pi) = \overline{V_\Pi}_{m_1} + \overline{\Psi(B_\Pi)}_{m_2} + c_1 + c_2$$

which hold with probability $1-\delta_1-\delta_2$, *where* $c_1 = \sqrt{\frac{\Psi(B)^2\log(2/\delta_1)}{2m_1}}$, $c_2 = 2\Psi(B)\sqrt{\frac{1}{m_2}\log\left(\frac{8n}{\delta m_2}\right)}$.

This upper bound depends on n, the maximum number of samples of $\Psi(B_\Pi)$. For any policy Π, the minimum width of a confidence interval of $\Psi(B_\Pi)$ required by the algorithm `BoundValueShare` when run with precision parameter ϵ is $\epsilon/4$. Hence, taking,

$$n = \left\lceil \frac{16^2\Psi(B)^2\log(8/\delta)}{\epsilon^2} \right\rceil, \tag{2}$$

ensures that for all policies, $2c_2 \leq \epsilon/4$ when $m_2 = n$. As discussed in Section 4, this is a necessary condition for the termination of the algorithm.

4 Algorithm

The process of sampling the reward of a policy involves sampling item sizes to decide which item to play next. We propose to make better use of all available data by using the samples of item sizes to calculate $U(\Psi(B_\Pi))$. Our algorithm, `OpStoK`, will then use the tight upper bound $U(\Psi(B_\Pi))$ in the bound future rewards, $U(V^+_\Pi)$. We also pool samples of the reward and size of items across policies, thus reducing the number of calls to the generative model. `OpStoK` also benefits from an alternative sampling method that reduces sample complexity and ensures that an entire ϵ-optimal policy is returned when the algorithm stops (line 5, Algorithm 1). This is achieved by using the bound in Proposition 1 and n as defined in (2).

In the main algorithm `OpStoK` (Algorithm 1), is very similar to `StOP-K` from [16] with the key differences appearing in the sampling and construction of confidence bounds, defined in `BoundValueShare`, that ensure the algorithm converges. `OpStoK` proceed by maintaining a set

Algorithm 1: `OpStoK` $(I, \delta_{0,1}, \delta_{0,2}, \epsilon)$

Initialization: $\textsc{Active} = \emptyset$
forall the $i \in I$ **do**
  $\Pi_i =$ policy consisting of just playing item i.
  $d(\Pi_i) = 1$
  $\delta_{1,1} = \frac{\delta_{0,1}}{d^*} N_1^{-1} \quad \delta_{1,2} = \frac{\delta_{0,2}}{d^*} N_1^{-1}$
  $(L(V^+_{\Pi_i}), U(V^+_{\Pi_i})) =$ `BoundValueShare` $(\Pi_i, \delta_{0,1}, \delta_{0,2}, \mathcal{S}^*, \epsilon)$
  $\textsc{Active} = \textsc{Active} \cup \{\Pi_i\}$.
end
for $t = 1, 2, \dots$ **do**
  $\Pi_t^{(1)} = \arg\max_{\Pi \in \text{Active}} U(V^+_\Pi)$
  $\Pi_t^{(2)} = \arg\max_{\Pi \in \text{Active} \setminus \{\Pi_t^{(1)}\}} U(V^+_\Pi)$
  if $L(V^+_{\Pi_t^{(1)}}) + \epsilon \geq \max_{\Pi \in \textsc{Active}} U(V^+_\Pi)$ **then**
    Stop: $\Pi^* = \Pi_t^{(1)}$;
  $\Pi_t = \Pi_t^{(a^*)}$, where $a^* = \arg\max_{a \in \{1,2\}} U(\Psi(B_{\Pi_t^{(a)}}))$
  $\textsc{Active} = \textsc{Active} \setminus \{\Pi_t\}$
  forall the *children* Π' *of* Π_t **do**
    $d(\Pi') = d(\Pi_t) + 1$
    $\delta_1 = \frac{\delta_{0,1}}{d^*} N^{-1}_{d(\Pi')}$ and $\delta_2 = \frac{\delta_{0,2}}{d^*} N^{-1}_{d(\Pi')}$
    $(L(V^+_{\Pi'}), U(V^+_{\Pi'})) =$ `BoundValueShare` $(\Pi', \delta_1, \delta_2, \mathcal{S}^*, \epsilon)$
    $\textsc{Active} = \textsc{Active} \cup \{\Pi'\}$
  end
end

of 'active' policies. As in [16] and [10], at each time step t, a policy, Π_t to expand is chosen by comparing the upper confidence bounds of the two best active policies. We select the policy with most uncertainty in the bounds since we want to be confident enough in our estimates of the near-optimal policies to say that the policy we ultimately select is better (see Figure 1). Once we have selected a policy, Π_t, if the stopping criteria is not met, we replace Π_t in the set of active policies with all its children. For each child policy, we use `BoundValueShare` to bound its reward. In order for all our bounds to hold simultaneously with probability greater than $1 - \delta_{0,1} - \delta_{0,2}$, `BoundValueShare` must be called with parameters

$$\delta_{d,1} = \frac{\delta_{0,1}}{d^*} N^{-1}_{d(\Pi)} \quad \text{and} \quad \delta_{d,2} = \frac{\delta_{0,2}}{d^*} N^{-1}_{d(\Pi)} \tag{3}$$

where N_d is the number of policies of depth d as given in (1). Our algorithm, `OpStoK` is given in Algorithm 1. The algorithm relies on `BoundValueShare` and subroutines, `EstimateValue` and `SampleBudget`, which sample the reward and budget of policies.

In `BoundValueShare`, we use samples of both item size and reward to bound the value of a policy. We define upper and lower bounds on the value of any extension of a policy Π as,

$$U(V^+_\Pi) = \overline{V_\Pi}_{m_1} + \overline{\Psi(B_\Pi)}_{m_2} + c_1 + c_2,$$
$$L(V^+_\Pi) = \overline{V_\Pi}_{m_1} - c_1,$$

with c_1 and c_2 as in Proposition 1. It is also possible to define upper and lower bounds on $\Psi(B_\Pi)$ with m_2 samples and confidence δ_2. From this, we can formally define a *complete* policy as a policy Π with $U(B_\Pi) = \overline{\Psi(B_\Pi)}_{m_2} + c_2 \leq \frac{\epsilon}{2}$. For complete policies, since there is very little capacity left, it is more important to get tight confidence bounds on the value of the policy. Hence, in `BoundValueShare`, we sample the remaining budget policy as much as is necessary to conclude whether the policy is complete or not. As soon as we realize we have a complete policy ($U(B_\Pi) \leq \epsilon/2$), we sample the value sufficiently to get a confidence interval of width less than ϵ. Then, when it comes to choosing an optimal policy to return, the confidence intervals of all complete policies will be narrow enough for this to happen. This is appropriate since, pre-specifying the number of samples may not lead

Algorithm 2: `BoundValueShare`$(\Pi, \delta_1, \delta_2, S^*, \epsilon)$

Input : Π: policy; δ_1: probability capacity confidence bound fails; δ_2: probability reward confidence bound fails; S^*: observed samples for all items; ϵ: tolerated approximation error.

Initialization : For all $i \in I$, let $\mathcal{S}_i = \mathcal{S}_i^*$

1 Set $m_2 = 1$ and $(\psi_1, \mathcal{S}) =$ `SampleBudget`$(\Pi, \mathcal{S})$

`/* draw a sample of the remaining budget */`

2 $\overline{\Psi(B_\Pi)}_{m_2} = \frac{1}{m_2} \sum_{j=1}^{m_2} \psi_j$

3 $U(\Psi(B_\Pi)) = \overline{\Psi(B_\Pi)}_{m_2} + 2\Psi(B)\sqrt{\frac{1}{m_2} \log\left(\frac{8n}{\delta m_2}\right)}$,

$L(\Psi(B_\Pi)) = \overline{\Psi(B_\Pi)}_{m_2} - 2\Psi(B)\sqrt{\frac{1}{m_2} \log\left(\frac{8n}{\delta m_2}\right)}$

`/* calculate upper and lower bounds on the remaining budget */`

4 **if** $U(\Psi(B_\Pi)) \leq \frac{\epsilon}{2}$ **then** $m_1 = \left\lceil \frac{8\Psi(B)^2 \log(2/\delta_1)}{\epsilon^2} \right\rceil$;

5 **else if** $L(\Psi(B_\Pi)) \geq \frac{\epsilon}{4}$ **then**

6 $\quad m_1 = \left\lceil \frac{1}{2} \frac{\Psi(B)^2 \log(2/\delta_1)}{u(\Psi(B))^2} \right\rceil$

7 **else**

8 $\quad$ Set $m_2 = m_2 + 1$, $(\psi_{m_2}, \mathcal{S}) =$ `SampleBudget`$(\Pi, \mathcal{S})$ and go back to 2

9 $\overline{V_\Pi}_{m_1} =$ `EstimateValue`(Π, m_1)

10 $L(V_\Pi^+) = \overline{V_\Pi}_{m_1} - \sqrt{\frac{\Psi(B)^2 \log(2/\delta_1)}{2m_1}}$

11 $U(V_\Pi^+) = \overline{V_\Pi}_{m_1} + \overline{\Psi(B_\Pi)}_{m_2} + \sqrt{\frac{\Psi(B)^2 \log(2/\delta_1)}{2m_1}} + 2\Psi(B)\sqrt{\frac{1}{m_2} \log\left(\frac{8n}{\delta m_2}\right)}$

12 **return** $(L(V_\Pi^+), U(V_\Pi^+))$

to confidence bounds tight enough to select an ϵ-optimal policy. If a complete policy is chosen as $\Pi_t^{(1)}$ in `OpStoK`, for some t, the algorithm will stop and this policy will be returned. For this to happen, we also need the stopping criterion to be checked before selecting a policy to expand. Note that in `BoundValueShare`, the reward and remaining budget must be sampled separately as we are considering closed-loop planning so the item chosen may depend on the size of the previous item, and hence the reward will depend on the instantiated item sizes. In line 6 of `BoundValueShare`, the number of samples of the reward, m_1, is defined to ensure that the uncertainty in the estimate of V_Π is less than $u(\Psi(B)) = \min\{U(\Psi(B_\Pi)), \Psi(B)\}$, since a natural upper bound for the reward is $\Psi(B)$. In the other case (when $U(\Psi(B_\Pi) \leq \epsilon/2$), we define it to ensure the confidence bounds are tight enough.

`OpStoK`, considerably reduces the number of calls to the generative model by creating sets $\mathcal{S}_i^*$ of samples of the reward and size of each item $i \in I$. When it is necessary to sample the reward and size of an item for the evaluation of a policy, we sample without replacement from $\mathcal{S}_i^*$, until $|\mathcal{S}_i^*|$ samples have been taken. At this point new calls to the generative model are made and the new samples added to the sets for use by future policies. We denote by $\mathcal{S}^*$ the collection of all sets $\mathcal{S}_i^*$.

4.1 Analysis

We state the following result guaranteeing the performance of `OpStoK`

Proposition 2 *With probability at least* $(1 - \delta_{0,1} - \delta_{0,2})$*, the algorithm* `OpStoK` *returns an action with value at least* $v^* - \epsilon$ *for* $\epsilon > 0$.

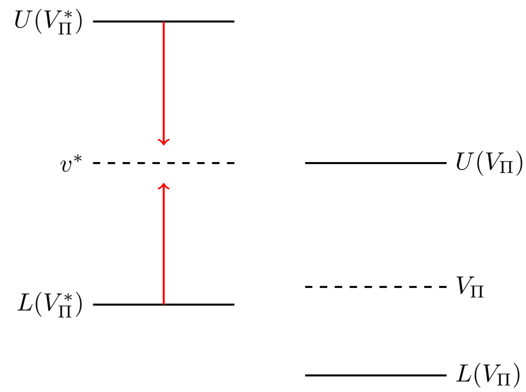

Figure 1: Example of where just looking at the optimistic policy might fail: If we always play the optimistic policy, as $U(V^+_{\Pi^*}) \geq U(V^+_\Pi)$, we will always play Π^* so the confidence bounds on Π will not shrink. This means that $L(V^+_{\Pi^*})$ will never be (epsilon) greater than the best alternative upper bound so there will not be enough confidence to conclude we have found the best policy.

5 Applications to online education

Our algorithm as discussed here has been in a mainly theoretical framework. However, we believe that it can have serious practical impact in the field of online education. For the problem of determining which question to give to a user in a fixed time frame, the tree based structure to question selection feels very natural. It allows for pre-requisite exercises to be included very easily, which along with producing more pedagogically sound policies will also improve computational efficiency as the search space will be reduced at the policy expansion stage. With this in mind, it may also be possible to incorporate ideas from the Zone of Proximal Development into the algorithm. Additionally, it would be interesting to consider reward which change depending on which questions have previously been asked. `OpStoK` provides an intuitive manner of dealing with the large decision trees that can arise when considering the problem of question selection in online education. Our algorithm will only evaluate potentially optimal policies and can be run offline (provided a good model of student performance and time are available) to produce educational policies that adapt to the remaining time and provide a near optimal learning experience for the limited period of time the user has to spend on the app or platform.

6 Conclusion

In this paper we have presented a new algorithm `OpStoK` that applies the optimistic planning strategy to the stochastic knapsack problem. This algorithm is directly motivated by problem of selecting an adaptive sequence of questions to give to a student in an education app. At present, we have provided largely theoretical results so in future it would be good to investigate the performance of our algorithm in a real education environment. While it is suspected that the algorithm will be computationally efficient as it reduces the number of policies that have to be explored, it would be interesting to run some experiments both to investigate its practical performance, both in artificial environments and the true educational setting for which it was designed.

References

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[2] K. Azuma. Weighted sums of certain dependent random variables. *Tohoku Math. J. (2)*, 1967.

[3] S. Bubeck and R. Munos. Open loop optimistic planning. In *Conference on Learning Theory*, pages 477–489, 2010.

[4] L. Busoniu and R. Munos. Optimistic planning for markov decision processes. In *15th International Conference on Artificial Intelligence and Statistics*, volume 22, pages 182–189, 2012.

[5] M. Chi, K. VanLehn, D. Litman, and P. Jordan. Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. *User Modeling and User-Adapted Interaction*, 21(1-2):137–180, 2011.

[6] B. Clement, D. Roy, P.-Y. Oudeyer, and M. Lopes. Multi-armed bandits for intelligent tutoring systems. *arXiv preprint arXiv:1310.3174*, 2013.

[7] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[8] G. B. Dantzig. Discrete-variable extremum problems. *Operations Research*, 5(2):266–288, 1957.

[9] B. C. Dean, M. X. Goemans, and J. Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Mathematics of Operations Research*, 33(4):945–964, 2008.

[10] V. Gabillon, M. Ghavamzadeh, and A. Lazaric. Best arm identification: A unified approach to fixed budget and fixed confidence. In *Advances in Neural Information Processing Systems*, pages 3212–3220, 2012.

[11] R. K. Hambleton. *Fundamentals of item response theory*. Sage publications, 1991.

[12] J.-F. Hren and R. Munos. Optimistic planning of deterministic systems. In *Recent Advances in Reinforcement Learning*, pages 151–164. Springer, 2008.

[13] J. I. Lee and E. Brunskill. The impact on individualizing student models on necessary practice opportunities. *International Educational Data Mining Society*, 2012.

[14] V. Perchet, P. Rigollet, S. Chassang, and E. Snowberg. Batched bandit problems. *The Annals of Statistics*, 44(2):660–681, 2016.

[15] A. N. Rafferty, E. Brunskill, T. L. Griffiths, and P. Shafto. Faster teaching via pomdp planning. *Cognitive Science*, 2015.

[16] B. Szörényi, G. Kedenburg, and R. Munos. Optimistic planning in markov decision processes using a generative model. In *Advances in Neural Information Processing Systems*, pages 1035–1043, 2014.

[17] W. J. van der Linden. A lognormal model for response times on test items. *Journal of Educational and Behavioral Statistics*, 31(2):181–204, 2006.

[18] D. Williams. *Probability with martingales*. Cambridge university press, 1991.

Proof of Proposition 1

The proof of Proposition 1 relies on the following key resul.t

Lemma 3 *Let $\{Z_m\}_{m=1}^{\infty}$ be a martingale with Z_m defined on the filtration $\mathcal{F}_m$, $E[Z_m] = 0$ and $|Z_m - Z_{m-1}| \leq d$ for all m where $Z_0 = 0$. Then,*

$$P\left(\exists m \leq n; \frac{Z_m}{m} \geq 2d^2\sqrt{\frac{2}{m}\log\left(\frac{n}{m}\frac{4}{\delta}\right)}\right) \leq \delta$$

Proof: The proof is similar to that of Lemma B.1 in [14] and will make use of the following standard results:

Theorem 4 Doob's maximal inequality: *let Z be a non-negative submartingale. Then for $c > 0$,*

$$P\left(\sup_{k\leq n} Z_k \geq c\right) \leq \frac{E[Z_n]}{c}.$$

Proof: See, for example, [18], Theorem 14.6, page 137. □

Lemma 5 *Let Z_n be a martingale such that $|Z_i - Z_{i-1}| \leq d_i$ for all i with probability 1. Then, for $\lambda > 0$,*

$$E[e^{\lambda Z_n}] \leq e^{\frac{\lambda^2 D^2}{2}},$$

where $D^2 = \sum_{i=1}^{n} d_i^2$.

Proof: See the proof of the Azuma-Hoeffding inequality in [2]. □

Then, for the proof or Lemma 3, we first notice that since $\{Z_m\}_{m=1}^{\infty}$ is a martingale, by Jensen's inequality for conditional expectations, it follows that for any $\lambda > 0$,

$$E[e^{\lambda Z_m}|\mathcal{F}_{m-1}] \geq e^{\lambda E[Z_m|\mathcal{F}_{m-1}]} = e^{\lambda Z_{m-1}}.$$

Hence, for any $\lambda > 0$, $\{e^{\lambda Z_m}\}_{m=1}^{\infty}$ is a positive sub-martingale so we can apply Doob's maximal inequality (Theorem 4) to get

$$P\left(\sup_{m\leq n} Z_m \geq c\right) = P\left(\sup_{m\leq n} e^{\lambda Z_m} \geq e^{\lambda c}\right) \leq \frac{E[e^{\lambda Z_n}]}{e^{\lambda c}}.$$

Then, by Lemma 5, since $|Z_i - Z_{i-1}| \leq d$ for all i, it follows that

$$P\left(\sup_{m\leq n} Z_m \geq c\right) \leq \frac{E[e^{\lambda Z_n}]}{e^{\lambda c}} \leq \frac{e^{\lambda^2 D^2/2}}{e^{\lambda c}} = \exp\left\{\frac{\lambda^2 D^2}{2} - \lambda c\right\}. \tag{4}$$

Minimizing the right hand side with respect to λ gives $\hat{\lambda} = \frac{c}{D^2}$ and substituting this back into (4) we get,

$$P\left(\sup_{m\leq n} Z_m \geq c\right) \leq \exp\left\{-\frac{c^2}{2D^2}\right\}.$$

Then, since we are considering the case where $d_i = d$ for all i, $D^2 = nd^2$ and so,

$$P\left(\sup_{m\leq n} Z_m \geq c\right) \leq \exp\left\{-\frac{c^2}{2nd^2}\right\}.$$

Further, if we are interested in $P(\sup_{k\leq m\leq n} Z_m \geq c)$, we can redefine the indices's to get

$$P\left(\sup_{k\leq m\leq n} Z_m \geq c\right) = P\left(\sup_{m'\leq n-k+1} Z_m \geq c\right) \leq \exp\left\{-\frac{c^2}{2(n-k+1)d^2}\right\}. \tag{5}$$

We then define $\varepsilon_m = 2d\sqrt{\frac{1}{m}\log\left(\frac{n}{m}\frac{8}{\delta}\right)}$ and use a peeling argument similar to that in Lemma B.1 of [14] to get

$$\begin{aligned}
P\left(\exists m \leq n; \frac{Z_m}{m} \geq \varepsilon_m\right) &\leq \sum_{t=0}^{\lfloor \log_2(n) \rfloor + 1} P\left(\bigcup_{m=2^t}^{2^{t+1}-1}\left\{\frac{Z_m}{m} \geq \varepsilon_m\right\}\right) && \text{(by union bound)}\\
&\leq \sum_{t=0}^{\lfloor \log_2(n) \rfloor + 1} P\left(\bigcup_{m=2^t}^{2^{t+1}-1}\left\{\frac{Z_m}{m} \geq \varepsilon_{2^{t+1}}\right\}\right) && \text{(since } \varepsilon_m \text{ decreasing in } m\text{)}\\
&\leq \sum_{t=0}^{\lfloor \log_2(n) \rfloor + 1} P\left(\bigcup_{m=2^t}^{2^{t+1}-1}\left\{Z_m \geq 2^t \varepsilon_{2^{t+1}}\right\}\right) && \text{(as } m \geq 2^t\text{)}\\
&\leq \sum_{t=0}^{\lfloor \log_2(n) \rfloor + 1} \exp\left\{-\frac{(2^t\varepsilon_{2^{t+1}})^2}{2^{t+1}d^2}\right\} && \text{(from (5))}\\
&\leq \sum_{t=0}^{\lfloor \log_2(n) \rfloor + 1} \frac{2^{t+1}\delta}{8n} && \text{(substituting } \varepsilon_{2^{t+1}}\text{)}\\
&\leq \frac{2^{\log_2(n)+3}\delta}{8n} = \delta. && \text{(since } \sum_{i=1}^{k} 2^i = 2^{k+1} - 1\text{)}
\end{aligned}$$

□

We are now able to prove the following proposition.

Proposition 6 (Proposition 1 in main text) *The Algorithm `BoundValueShare` (Algorithm 2) returns confidence bounds,*

$$L(V_\Pi) = \overline{V_\Pi}_{m_1} - \sqrt{\frac{\Psi(B)^2\log(2/\delta_1)}{2m_1}}$$

$$U(V_\Pi) = \overline{V_\Pi}_{m_1} + \overline{\Psi(B_\Pi)}_{m_2} + \sqrt{\frac{\Psi(B)^2\log(2/\delta_1)}{2m_1}} + 2\Psi(B)\sqrt{\frac{1}{m}\log\left(\frac{8n}{\delta m}\right)}$$

which hold with probability $1 - \delta_1 - \delta_2$.

Proof:

We then begin by noting that our samples of item size are dependent because we evaluate in each iteration a bound based on past samples and we use this bound to decide if we need to continue sampling or if we can stop. To model this dependence let us introduce a stopping time τ such that $\tau(\omega) = n$ if our algorithm exits the loop at n. Consider the sequence

$$\overline{\Psi(B_\Pi)}_{1\wedge\tau}, \overline{\Psi(B_\Pi)}_{2\wedge\tau}, \ldots$$

and define for $m \geq 1$

$$M_m = (m \wedge \tau)(\overline{\Psi(B_\Pi)}_{m\wedge\tau} - E[\Psi(B_\Pi)]) \quad \text{with} \quad M_0 = 0.$$

Furthermore, define the filtration $\mathcal{F}_m = \sigma(B_{\Pi,1}, \ldots, B_{\Pi,m})$ then for $m \geq 1$

$$E[M_m|\mathcal{F}_{m-1}] = E[M_m|\mathcal{F}_{m-1}, \tau \leq m-1] + E[M_m|\mathcal{F}_{m-1}, \tau > m-1].$$

Now

$$E[M_m|\mathcal{F}_{m-1}, \tau \leq m-1] = E[M_{m-1}|\tau \leq m-1].$$

and due to independence of the samples $B_{\Pi,1}, \ldots, B_{\Pi,m}$

$$
\begin{aligned}
&E[M_m|\mathcal{F}_{m-1}, \tau > m-1] \\
&= E[m(\overline{\Psi(B_\Pi)}_m - E[\Psi(B_\Pi)])|\mathcal{F}_{m-1}, \tau > m-1] \\
&= E\left[\sum_{j=1}^{m-1} \Psi(B_{\Pi,j}) + \Psi(B_{\Pi,m}) - mE[\Psi(B_\Pi)]\middle|\mathcal{F}_{m-1}, \tau > m-1\right] \\
&= (m-1)E[\overline{\Psi(B_\Pi)}_{m-1} - E[\Psi(B_\Pi)]|\mathcal{F}_{m-1}, \tau > m-1] \\
&\qquad\qquad\qquad\qquad + E[\Psi(B_{\Pi,m}) - E[\Psi(B_\Pi)]|\mathcal{F}_{m-1}, \tau > m-1] \\
&= E[M_{m-1}|\tau > m-1] + E[\Psi(B_{\Pi,m})] - E[\Psi(B_\Pi)] = E[M_{m-1}|\tau > m-1].
\end{aligned}
$$

Hence, $E[M_m|\mathcal{F}_{m-1}] = M_{m-1}$ and M_m is a martingale with increments $|M_m - M_{m-1}| \leq |\Psi(B_{\Pi,m}) - E[\Psi(B_\Pi)]| \leq \Psi(B)$. We could apply the Azuma-Hoeffding inequality to gain guarantees for individual m-values. Alternatively, we can use Lemma 3 to get,

$$
P\left(\sup_{m\leq n} \frac{M_m}{m} \geq 2\Psi(B)\sqrt{\frac{1}{m}\log\left(\frac{8n}{\delta m}\right)}\right) \leq \delta_2.
$$

Using this in conjunction with Azuma bounds on the reward of a policy gives

$$
\overline{V_\Pi}_{m_1} - c_1 \leq V_\Pi^+ \leq \overline{V_\Pi}_{m_1} + \overline{\Psi(B_\Pi)}_{m_2} + c_1 + c_2,
$$

where $c_1 := \sqrt{\frac{\Psi(B)^2 \log(2/\delta_1)}{2m_1}}$ and $c_2 := 2\Psi(B)\sqrt{\frac{1}{m}\log\left(\frac{8n}{\delta m}\right)}$ and these bounds hold with probability $1 - \delta_1 - \delta_2$.

$\square$